LAHDEN AMMATTIKORKEAKOULU
Lahti University of Applied Sciences

MULTIDOMAIN CMS MADE SIMPLE

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikka
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2013
Jari Hyle

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

HYLE, JARI: Multidomain CMS Made Simple

Ohjelmistotekniikan opinnäytetyö, 33 sivua

Kevät 2013

TIIVISTELMÄ

Tämä opinnäytetyö käsittelee sisällönhallintajärjestelmän muokkaamista usealla domainilla toimivaksi. Näin yhdellä sisällönhallintajärjestelmän asennuksella saadaan luotua useita sivustoja.

Työn teoriaosuudessa käydään läpi hieman toimintaympäristöä ja sisällönhallintajärjestelmiä yleisesti. Tämän jälkeen tutustutaan nimipalvelujärjestelmän toimintaan ja itse työssä käytettyyn CMS Made Simple -nimiseen sisällönhallintajärjestelmään.

CMS Made Simple on avoimen lähdekoodin sisällönhallintajärjestelmä. Se käyttää Smarty template-engineä, joka prosessoi sivupohjat ja toimii muutenkin sisällönhallintajärjestelmän moottorina taustalla.

Toteutus luotiin pH Kolme -yrityksen alaisuudessa. Projekti oli prototyyppinen ja sitä on tarkoitus myydä asiakkaille järjestelmänä, jossa asiakas voi luoda sivustoja loppukäyttäjälle.

Toteutus tehtiin vaatimusten mukaiseksi ja sitä muokattiin asiakkaan palautteen myötä. Lopputuloksena useita sivustoja saatiin liitettyä CMS Made Simplen yhteen asennukseen ja loppukäyttäjä oli tyytyväinen tulokseen.

Asiasanat: CMS Made Simple, PHP, Smarty, Web-sivut, domain

Lahti University of Applied Sciences
Degree Programme in Information Technology

HYLE, JARI: Multidomain CMS Made Simple

Bachelor's Thesis in software engineering, 33 pages
Spring 2013

ABSTRACT

This Bachelor's thesis deals with modifying a content management system to work with several domains. This modification enables one installation of a content management system to create multiple sites.

The theory part of the thesis explains some of the operational environment and content management systems in general. After this the operation of the domain name system and a content management system known as CMS Made Simple, which is featured in this thesis, are introduced.

CMS Made Simple is an open source content management system. It utilizes the Smarty template engine, which processes the templates and works as an engine for the content management system on the background.

The practical case was done under the supervision of pH Kolme company. The project was prototypical in nature and it is meant to be sold to customers as a system where the customer is able to create sites for the end user.

The execution was made to meet the demands and altered according to the customers' feedback. As a result, several sites were attached to a single installation of CMS Made Simple and the end user was pleased with the result.

Key words: CMS Made Simple, PHP, Smarty, Web-sites, domain

SISÄLLYS

1 JOHDANTO

Internetpalvelujen kehittyessä on markkinoille ilmestynyt monta erilaista sisällönhallintajärjestelmätyökalua, joilla on aloittelijankin helppo luoda internetsivut ilman suurempaa vaivaa. Harva ilmainen sisällönhallintajärjestelmä on kuitenkaan erikoistunut useamman kuin yhden sivuston hallintaan per sisällönhallintajärjestelmän asennus.

pH Kolme on pieni ohjelmistoyritys Lahdesta, jonka liikevaihto on noin 113 tuhatta euroa. Se on perustettu vuonna 2002 ja on monipuolinen ohjelmistotalo, joka on viime vuosina keskittynyt enemmän mediapuolen ohjelmistoihin.

Tämä opinnäytetyö on tehty pH Kolme -yrityksen alaisuudessa. Työ tehtiin prototyyppisenä projektina, ja tavoitteena oli luoda usealla domainilla toimiva sisällönhallintajärjestelmän asennus. Työn toteuttamiseen käytettiin CMS Made Simple –nimistä sisällönhallintajärjestelmää asiakasvaatimusten takia.

Tutkimusongelmana työssä on ollut saada asennus hyvin geneeriseksi mutta samalla helposti muokattavaksi. Opinnäytetyössä käsitellään asiat lähinnä CMS Made Simplen näkökulmasta. Käytännön osuus pyrkii selittämään järjestelmän luomisen tavalla, joka täyttäisi mahdollisimman hyvin lukijan tarpeen luoda multidomain sivusto CMS Made Simple sisällönhallintajärjestelmällä. Työn teoriaosuudessa kerrotaan aluksi internetin nimipalvelusta ja myöhemmin tutustutaan CMS Made Simplen toimintaan.

2 TOIMINTAYMPÄRISTÖN KUVAUS

2.1 CMS

CMS tulee sanoista Content Management System eli suomeksi sisällönhallintajärjestelmä. Sisällönhallintajärjestelmä on tietojärjestelmä, joka pystyy hallitsemaan, sisällönhallintajärjestelmästä riippuen, määrättyä organisaation tai tarpeen aluetta. Näitä alueita ovat esimerkiksi dokumenttihallintajärjestelmä, julkaisujärjestelmä, www-sisällönhallintajärjestelmää ja verkkokauppajärjestelmä. (Wikipedia 2012a.)

Tässä opinnäytetyössä sisällönhallintajärjestelmällä tarkoitetaan www-sisällönhallintajärjestelmää, jolla pyritään hallitsemaan verkkopalvelun sisältöjä mahdollisimman tarkoituksenmukaisesti. Www-sisällönhallinnalle tyypillistä on sivupohjien käyttö, joka yhtenäistää sivustoa. Sivupohjissa saadaan eri sivuille toistumaan esimerkiksi navigaatio ja sivun yleinen muoto. (Wikipedia 2012c.)

Www-sisällönhallintajärjestelmää hallitaan yleensä selaimen avulla. Graafinen käyttöliittymä selaimella helpottaa sisällönhallinan käyttöä huomattavasti. Eri osiot ovat selkeästi eroteltuna, ja määrättyjen osioiden hallinta voidaan pilkkoa eri käyttäjille. (Wikipedia 2012b.)

2.1.1 Asennus

Www-sisällönhallintajärjestelmä vaatii www-palvelimen. Palvelimelta yleensä vaaditaan AMP (Apache, MySQL ja PHP) -ominaisuuksia. Käyttöjärjestelmä palvelimella on yleensä Linux-pohjainen (LAMP), mutta monet sisällönhallintajärjestelmät toimivat myöskin Windows-pohjaisilla (WAMP) AMP-palvelimilla.

Suosituimmissa ilmaisissa sisällönhallintajärjestelmissä asennustiedostot voi ladata suoraan sisällönhallintajärjestelmän kehittäjien www-sivuilta. Tiedostot ovat yleensä pakattuja yhteen tiedostoon. Tiedosto siirretään palvelimelle tiedostojärjestelmään ja puretaan kansioon, joka on näkyvissä internetiin. Jotkut

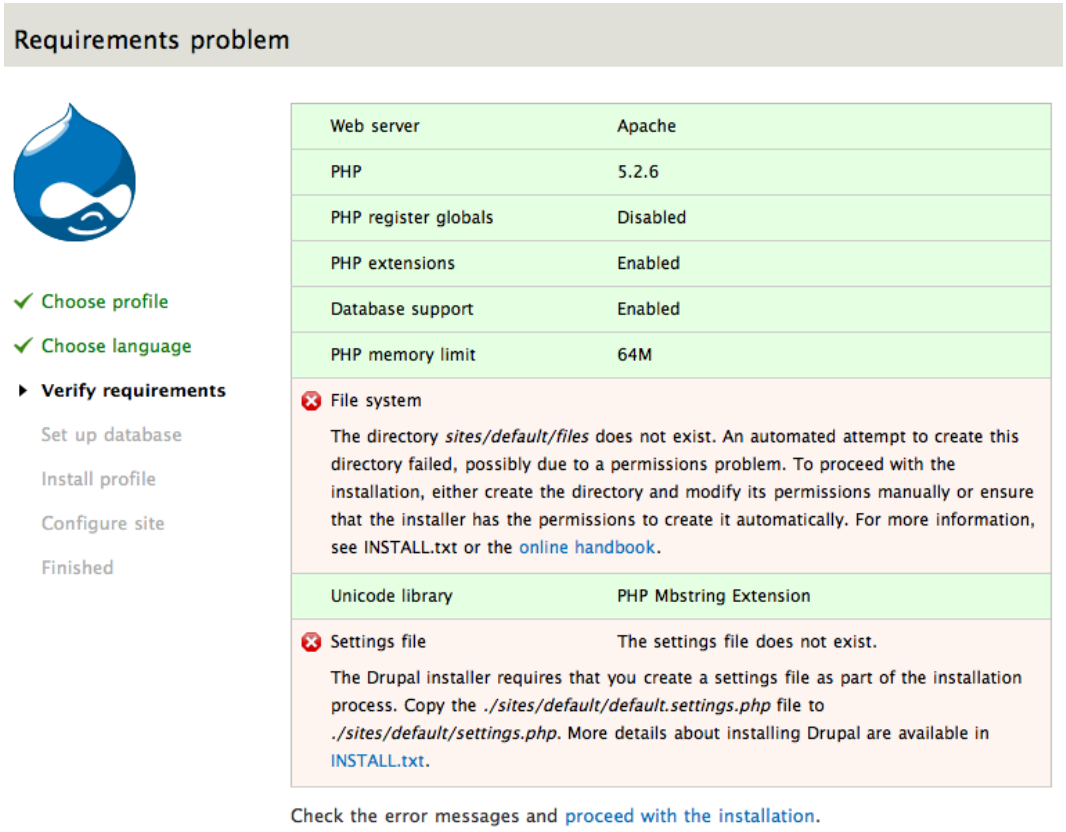
sisällönhallintajärjestelmät vaativat asetustiedoston ja kansion oikeuksien muuttamista niin, että asennus voi muokata niitä tarvittaessa.

Kuviossa 1 esitetään Drupalin asennukseen vaadittavia hakemistojärjestelmän tehtyjä komentoja. Ensimmäisellä rivillä haetaan asennustiedosto palvelimelle wget-komennolla. Toisella rivillä puretaan tämä tiedosto. Tämän jälkeen siirretään Drupalin pakatusta tiedostosta tulleita tiedostoja oikeille paikoille riveillä 3-4. Lopuksi poistetaan purettu kansio ja muokataan settings.php-tiedoston luku- ja kirjoitusoikeudet asennuksen vaatimille tasoille chmod-komennolla.

```
1 wget http://ftp.drupal.org/files/projects/drupal-x.x.tar.gz
2 tar -xzvf drupal-x.x.tar.gz
3 mv drupal-x.x/* drupal-x.x/.htaccess ./
4 mv drupal-x.x/.gitignore ./
5 rmdir drupal-x.x
6 chmod 666 sites/default/settings.php
```

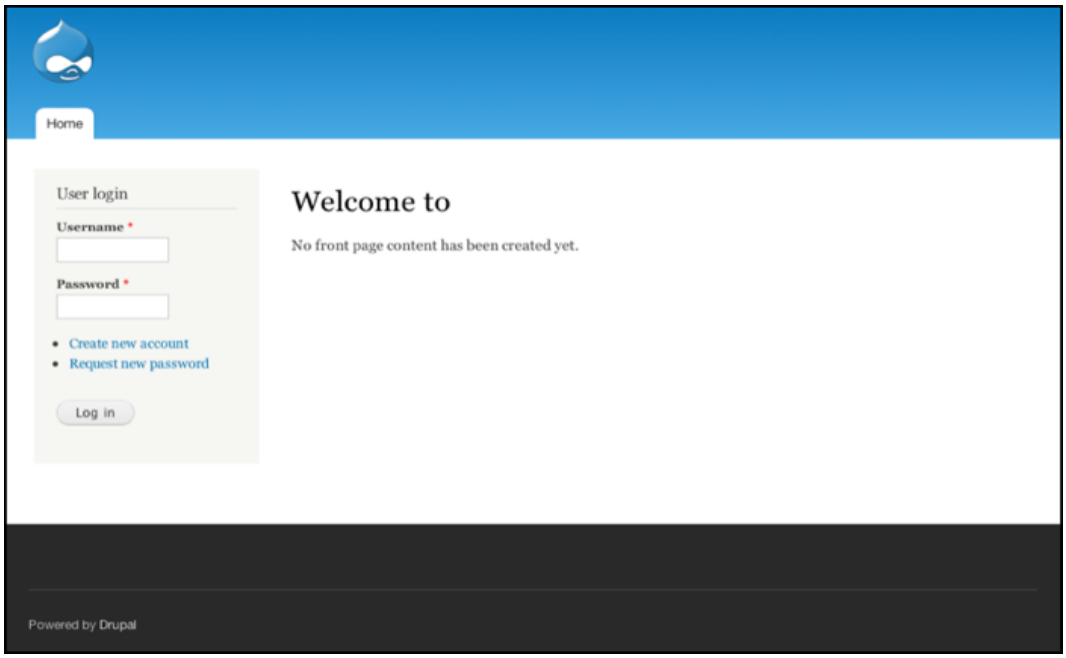
KUVIO 1. Esimerkki asennuskomennoista unix-ympäristössä

Asennuksen seuraavassa osassa ei enää kosketa tiedostojärjestelmiin, koska asennus yleensä suoritetaan selainpohjaisesti. Käyttäjä aukaisee selaimella asennustiedoston, joka ohjatusti asentaa sisällönhallintajärjestelmän tiedostojärjestelmään. Asennus varmistaa palvelimen vaadituista ominaisuuksista, jotta sisällönhallintajärjestelmä toimii varmasti palvelimella, kuten kuviossa 2 on esitetty. Seuraavaksi asennus kysyy käyttäjältä vaadittuja tietoja, kuten tietokannan osoitetta ja salasanaa ja sivuston nimeä. Tässä vaiheessa luodaan pääkäyttäjän tunnukset, joilla on koko sisällönhallintajärjestelmän kaikki oikeudet. Tämän jälkeen asennusohjelma asentaa sisällönhallintajärjestelmän käyttäjän määrittelemillä asetuksilla ja ilmoittaa asennuksen onnistumisesta.



KUVIO 2. Drupal-asennuksen testaus (Drupal Installation 2013)

Asennuksen jälkeen sivustolla saattaa olla demosivusto esillä välittömästi. Esimerkiksi CMS Made Simplellä on hyvin monimuotoinen sivusto rakennettuna heti asennuksen jälkeen. Drupalilla taas muodostuu asennuksen jälkeen vain etusivu, kuten kuviossa 3 on esitetty. Tältä etusivulta voi kirjautua sivuston hallintapaneeliin. Hallintapaneelista hallitaan koko sivustoa ja sen toimintaa. Drupalissa hallintapaneelissa ei ole valmiiksi asennettuna paljon toiminnallisuutta perusasennuksessa, mutta siihen voi ladata useista kymmenistä tuhansista moduuleista halutut toiminnot tai luoda ne itse.

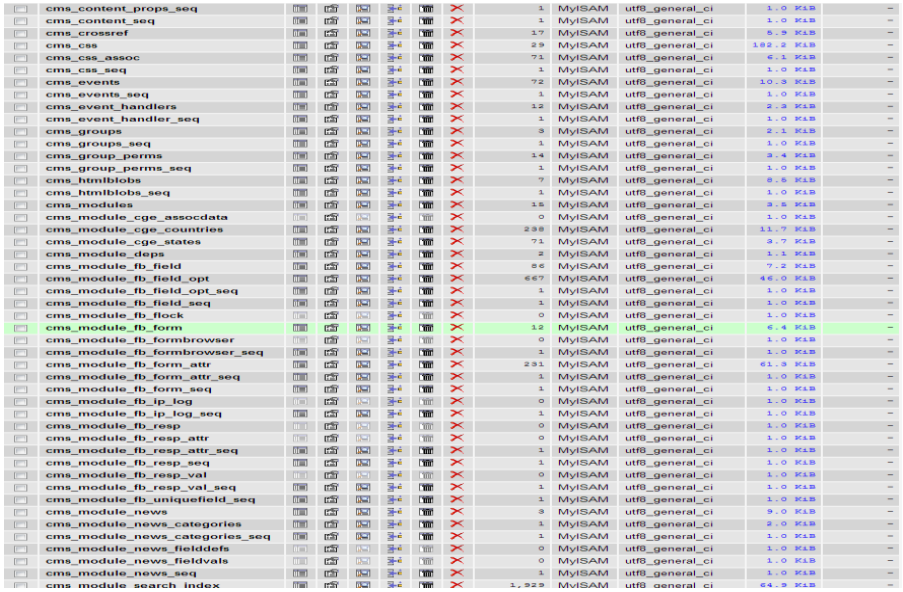


KUVIO 3. Drupal-etusivu asennuksen jälkeen

2.1.2 Rakenne ja toiminta

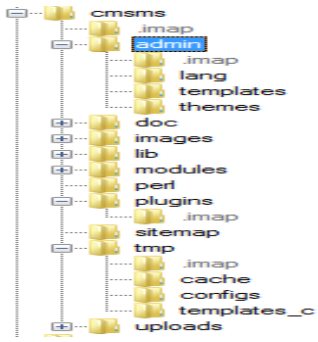
Www-sisällönhallintajärjestelmissä tärkeässä roolissa on tietokannan käyttö. Tietokantaan tallennetaan melkein kaikki sivuston tiedot. Näihin tietoihin luetaan esimerkiksi sivujen sisältö, sivupohjat ja tyylit. Sisällönhallintajärjestelmä luo sivuston vierailijalle sivut yhdistelemällä nämä tiedot sivupohjan määrittelemään muotoon. Osa sisällönhallintajärjestelmistä luo usein ladatuista sivustoista niin sanotut cache-tiedostot, jotka ovat valmiiksi sivupohjista ja sivujen sisällöstä luotuja tiedostoja. Tämä vähentää kyseisten sivujen latausaikoja.

Koska suurin osa tiedoista sisällönhallintajärjestelmissä tallennetaan tietokantaan, on tietokannan taulujen määrä hyvin suuri. Esimerkiksi kuviossa 4 on osa CMS Made Simplen tietokannan tauluista. Hakemistorakenteeseen jää kaikkien moduulien ja itse sisällönhallintajärjestelmän logiikka, joka hakee tiedot tietokannasta. CMS Made Simplen hakemistorakenne on esitetty kuviossa 5, jossa eri sisällönhallintajärjestelmän osat ovat eri kansioissa. Esimerkiksi Admin-kansiossa on sisällönhallintajärjestelmän hallintaa koskevat tiedostot ja Images-kansiossa on järjestelmän käyttämät kuvatiedostot.



cms_content_props_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_content_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_crossref							17	MyISAM	utf8_general_ci	5.9 KiB	-
cms_css							29	MyISAM	utf8_general_ci	102.2 KiB	-
cms_css_assoc							71	MyISAM	utf8_general_ci	6.1 KiB	-
cms_css_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_events							72	MyISAM	utf8_general_ci	10.3 KiB	-
cms_events_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_event_handlers							12	MyISAM	utf8_general_ci	2.3 KiB	-
cms_event_handler_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_groups							9	MyISAM	utf8_general_ci	2.1 KiB	-
cms_groups_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_group_perms							14	MyISAM	utf8_general_ci	3.4 KiB	-
cms_group_perms_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_htmlblobs							7	MyISAM	utf8_general_ci	8.5 KiB	-
cms_htmlblobs_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_modules							15	MyISAM	utf8_general_ci	3.5 KiB	-
cms_module_cge_assocdata							0	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_cge_countries							238	MyISAM	utf8_general_ci	11.7 KiB	-
cms_module_cge_states							71	MyISAM	utf8_general_ci	3.7 KiB	-
cms_module_deps							2	MyISAM	utf8_general_ci	1.1 KiB	-
cms_module_fb_field							86	MyISAM	utf8_general_ci	7.2 KiB	-
cms_module_fb_field_opt							667	MyISAM	utf8_general_ci	46.0 KiB	-
cms_module_fb_field_opt_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_fb_field_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_fb_flock							0	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_fb_form							12	MyISAM	utf8_general_ci	6.4 KiB	-
cms_module_fb_formbrowser							0	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_fb_formbrowser_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_fb_form_attr							231	MyISAM	utf8_general_ci	61.3 KiB	-
cms_module_fb_form_attr_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_fb_form_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_fb_ip_log							0	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_fb_ip_log_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_fb_resp							0	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_fb_resp_attr							0	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_fb_resp_attr_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_fb_resp_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_fb_resp_val							0	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_fb_resp_val_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_fb_uniquefield_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_news							8	MyISAM	utf8_general_ci	9.0 KiB	-
cms_module_news_categories							1	MyISAM	utf8_general_ci	2.0 KiB	-
cms_module_news_categories_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_news_fielddefs							0	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_news_fieldvals							0	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_news_seq							1	MyISAM	utf8_general_ci	1.0 KiB	-
cms_module_search_index							1,329	MyISAM	utf8_general_ci	64.9 KiB	-

KUVIO 4. Esimerkki sisällönhallintajärjestelmän tietokannasta



KUVIO 5. Esimerkki sisällönhallintajärjestelmän hakemistorakenteesta

Logiikan ja sivupohjan prosessoinnin ennen tulostusta hoitaa template engine. Sen avulla erotetaan ohjelmalogiikka presentaatiosta. PHP-pohjaisissa sivuissa se tarkoittaa PHP-kielen erottamista HTML- ja CSS-kielistä. HTML-tiedoston sijaan käytetään .tpl-tiedostoja, siis niin sanottuja mallinnetiedostoja, jotka template engine prosessoi ohjelmalogiikan määrittelyjen mukaisesti ennen presentaation tulostamista. Template enginessä on yleensä myös omaa logiikkaa käytettävissä. Yksi suosituimmista PHP template engineistä on tällä hetkellä Smarty, josta kuviot 6-8 ovat esimerkkinä.

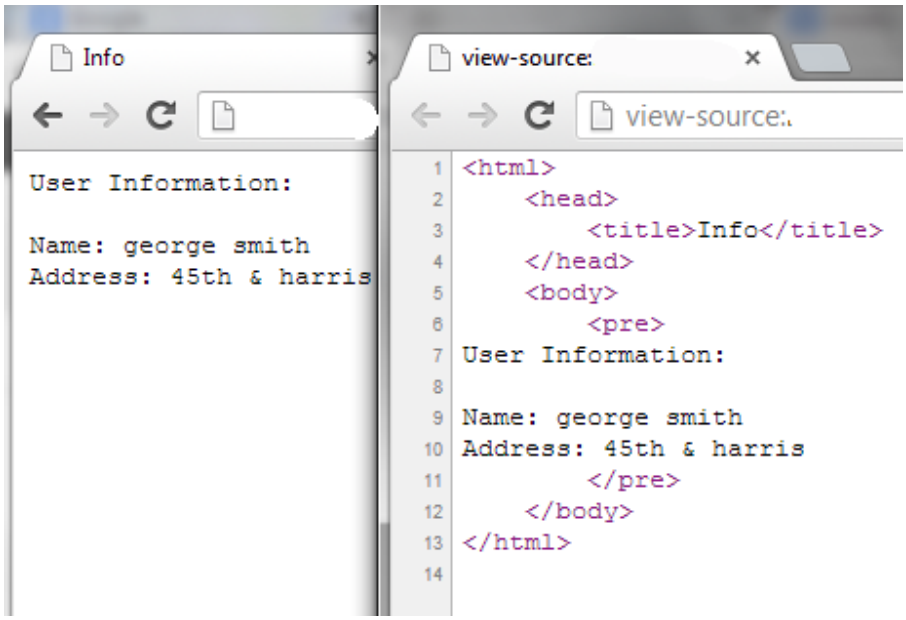
```
1  include('Smarty.class.php');
2  $smarty = new Smarty;
3  $smarty->assign('name', 'george smith');
4  $smarty->assign('address', '45th & Harris');
5  $smarty->display('index.tpl');
```

KUVIO 6. index.php-tiedosto, sivun toimintalogiikka

Kuviossa 6 käynnistetään Smarty, joka asettaa muuttujille ’name’ ja ’address’ arvot riveille 3 ja 4 ja rivillä 5 prosessoi ja tulostaa index.html-tiedoston index.tpl -tiedoston pohjalta. Kuviossa 7 on index.tpl-tiedosto, jossa Smartylle näytetään aaltosulkein $name- ja $address-muuttujien paikat riveillä 10 ja 11. Kun index.php avataan selaimessa, tulostuu selaimeen kuvion 8 vasemmassa laidassa olevat tekstit. Oikeassa laidassa kuviota on esitetty sivun lähdekoodi prosessoinnin jälkeen.

```
1  <html>
2  <head>
3  <title>Info</title>
4  </head>
5  <body>
6
7  <pre>
8  User Information:
9
10 Name: {$name}
11 Address: {$address}
12 </pre>
13
14 </body>
15 </html>
```

KUVIO 7. index.tpl-tiedosto, sivupohja



KUVIO 8. Sivun ulkoasu ja lähdekoodi prosessoinnin jälkeen

2.1.3 Käyttäjien ylläpito

Sisällönhallintajärjestelmissä on tapana jakaa ylläpitokäyttäjät eri tasoihin. Tämä mahdollistaa ylläpidon jakamista esimerkiksi organisaation eri osastoille. Näin eri osastot voivat keskittyä eri sivuston osiin, eikä heidän työhön koske kukaan luvaton. Sisällönhallintajärjestelmät mahdollistavat eri osa-alueiden osaamisen jakautumisen eri yritysten välille. Tämä mahdollistaa sen, että yritys voi myydä esimerkiksi valmiin moduulin, joka helpottaa sisällönhallintajärjestelmän käyttöä.

Käyttäjältä ei alimmilla tasoilla vaadita kuin tekstinkäsittelytaitoja hänen voidakseen toimia mukana sivuston ylläpidossa tai sisällön luomisessa. Tämä vapauttaa osaavampien käyttäjien resursseja, kun he voivat delegoida sisällön syöttämisen muille.

Sisällönhallintajärjestelmään on yleensä asennuksen jälkeen asentunut monta moduulia, jotka helpottavat sivuston ylläpidon eri osa-alueita. Näitä moduuleita on yleensä useita lisää sisällönhallintajärjestelmän omilla sivuilla. Moduulien avulla voi laajentaa sisällönhallintajärjestelmää halutulla tavalla. Esimerkiksi jokin moduuli voi auttaa luomaan sivustosta helposti verkkokaupan muutamalla klikkauksella.

2.1.4 Vertailu

Suuri osa PHP-pohjaisista järjestelmistä on lisensoitu avoimen lähdekoodin projekteiksi. Näin ollen suuri osa sisällönhallintajärjestelmistä on ilmaista ladata internetistä. Yleisimpiin järjestelmiin luetaan Drupal, Joomla! ja Wordpress. Tässä osiossa vertaillaan myöskin opinnäytetyössä käytettyä CMS Made Simpleä.

Kuviossa 9 on nämä sisällönhallintajärjestelmät vertailussa. ’Audit Trail’ tarkoittaa sisällön muokkaajien muokkauksien seuraamista. Sandboxilla tarkoitetaan hallintatyökalun kykyä sallia uusien sivuston ominaisuuksien testaamista sivustolla vaikuttamatta muuhun sivustoon.

	CMS Made Simple 1.9.4	Drupal 7.12	Joomla! 2.5.4	WordPress 3.3.2
Last Updated	3/7/2011	2/16/2012	5/2/2012	5/29/2012
System Requirements	CMS Made Simple 1.9.4	Drupal 7.12	Joomla! 2.5.4	WordPress 3.3.2
☑ *Application Server*	Apache	Apache	CGI	blank
☑ *Database*	MySQL	MySQL	MySQL	MySQL
☑ *License*	Open Source	Open Source	Open Source	Open Source
☑ *Programming Language*	PHP	PHP	PHP	PHP
☑ *Audit Trail*	Yes	Yes	No	Limited
☑ *Sandbox*	Limited	No	No	Limited
☑ *Advanced Caching*	Yes	Yes	Yes	Free Add On
☑ *Page Caching*	Yes	Yes	Yes	Free Add On

KUVIO 9. Sisällönhallintajärjestelmien vertailutaulukko (CMS Matrix 2013)

Sisällönhallintajärjestelmien vertailuissa ei kuitenkaan kovin usein oteta huomioon sisällönhallintajärjestelmien helppokäyttöisyyttä tai asennuksen ja käyttöönoton helppoutta. CMS Made Simple esimerkiksi tähtää helppokäyttöisyyteen ja siihen, että pelkällä asennuksella ja muutamalla pienellä muokkauksella sivusto on käytännössä valmis. Monessa sisällönhallintajärjestelmässä on myöskin pinnan alla paljon enemmän, mikä johtaa monesti harhaan, kun katsoo ominaisuuksia listasta. Täten sisällönhallintajärjestelmää valittaessa voidaan esimerkiksi asentaa muutama eri järjestelmä ja testata nopeasti, mitkä ominaisuudet tuntuvat sopivilta juuri itselle.

2.2 Asiakasvaatimukset

Työn tavoitteena on tehdä prototyyppi pH Kolme -yritykselle. Työssä käytetään CMS Made Simple -sisällönhallintajärjestelmää, koska siitä on yrityksessä eniten kokemusta. Projektin tekeminen muilla sisällönhallintajärjestelmillä olisi ehkä helpompaa tai jopa tarpeetonta, mutta uuden järjestelmän opetteleminen ja vanhan rinnalle ottaminen on kannattamatonta yrityksen kannalta.

Työssä on tarkoituksena tehdä useaan domainiin liitetty CMS Made Simplen asennus. Tämä ei ole CMS Made Simplen perusominaisuus, vaan se pitää lisätä muuttamalla sisällönhallintajärjestelmän toimintaa. Sisällönhallintajärjestelmään lisätyt domainit ohjaavat CMS Made Simplen asennuksen eri alisivuihin, jotka näyttävät loppukäyttäjälle erillisiltä sivustoilta.

Kun prototyyppi on valmis, se olisi tarkoitus myydä käyttäjälle, joka luo sillä pienimuotoisia sivustoja loppukäyttäjille. Sivustojen luominen on oltava helppoa ja aluksi niiden luominen on onnistuttava ehkä yhdellä tai kahdella sivupohjalla. Sivustojen luominen ei kuitenkaan saa vaikeutua CMS Made Simpleen tehtyjen muokkausten takia. Nämä sivustot saavat oman osoitenimensä (domainin), joka antaa loppuasiakkaalle mielikuvan, että sivusto on täysin oma.

3 INTERNETIN NIMIPALVELUJÄRJESTELMÄ

DNS eli Domain Name System on Internetissä se, joka määrittelee, mihin IP-osoitteeseen tietokone ottaa yhteyttä, kun se kutsuu Internetistä jotain nimeltä. Ilman tätä palvelua kaikki laitteet joutuisivat kommunikoimaan pelkän IP-osoitteiden avulla. Tämä on kuitenkin ihmiselle vaikeasti muistettavaa, joten nimipalvelu antaa ihmiselle helpomman tavan ottaa yhteyttä toisiin laitteisiin. (Anttila 2000, 231 – 232.)

Ennen tähän palveluun ei ollut kuitenkaan tarvetta vaan sen hoiti esim. nykyisessä Windowsissakin oleva hosts-tiedosto. Hosts-tiedoston huono puoli oli kuitenkin se, että jokaisen uuden koneen liittyessä verkkoon, sille piti määrittää jokaisessa koneessa nimiosoite. Ennen 1980-lukua tämä oli vielä helppoa, koska uusien laitteiden verkkoon liitetty määrä kasvoi vain kymmenissä per kuukausi. Nykyisin laitteita tulee miljoona tai jopa useita per kuukausi. (Anttila 2000, 231 – 232.)

Suurenevan laitemäärän nimeämiseksi DNS esitettiin ensimmäisen kerran vuonna 1983. Sen liikenne vastaa nykyään noin yhtä prosenttia Internetin runkoverkon liikenteestä. (Anttila 2000, 231 – 232.)

3.1 Osoitehierarkia

Nimet DNS:ssä jaetaan puumaiseen hierarkiseen rakenteeseen. Tämä puumaisuus jaotellaan pisteillä, ja jokaisesta pisteestä lähtee useita haaroja. Kuviossa 10 on esimerkki laitteen nimien osista. Puun ylimmällä tasolla on ccTLD eli Country Code Top Level Domain. Tämä puun osa koostuu maakohtaisista nimikkeistä esimerkiksi Suomella on domainnimi ”.fi”. Toinen vaihtoehto ccTLD:lle on sTLD eli Standard Top Level Domain, joka oli aikaisemmin käytössä vain Yhdysvalloissa mutta sen käyttö on sittemmin vapautettu muuallakkin maailmassa. sTLD:stä esimerkkinä yleisimmät ”.com” ja ”.net”. (Anttila 2000, 233 – 234.)

Seuraavalla tasolla on SLD eli Second Level Domain, jossa on kaksi eri vaihtoehtoa. Ensimmäisessä on piirin märittelyä täsmentävä tyyppikomponenti. Tätä käytetään yleensä esim Iso-Britanniassa ”.co.uk” täsmentämään kaupallisia ”.uk”-päätteisiä nimiä. Toinen vaihtoehto on organisaatio tai tuotemerkki tai vastaava. Tätä alemmat tasot ovat organisaation alayksikkö ja laitteen nimi, kuten kuviossa 10 on esitetty.(Anttila 2000, 233 – 235.)

KUVIO 10. Esimerkki laitteen nimien osista (Anttila 2000, 233.)

3.2 Nimien toiminta

Vaikka kaikille laitteille voidaan antaa nimi, vaatii IP-protokollan toimiminen IP-osoitteen laitteelta. Tämä IP-osoite siis hankitaan DNS-palvelun kautta. DNS-palvelussa on eritasoisia DNS-palvelimia. Laitteen kysyessä osoitetta ensimmäinen vaihe on yleensä yrityksen oma DNS-palvelin tai oman palveluntarjoajan DNS-palvelin. Tällä DNS-palvelimella on oma välimuisti kaikista kyseilyistä ja se saattaa tietää määrättyjen nimien osoitteet ilman lisäkyselyitä, jotta verkkoa ei tarvitse kuormittaa samoilla kyselyillä vähän väliä. Jos tämä DNS-palvelin ei tiedä kysyttyä osoitetta nimelle, se voi kysyä nimen sTLD tai ccTLD osan DNS-palvelimen osoitetta Root DNS-palvelimelta. Root DNS kertoo laitteen omalle DNS-palvelimelle osoitteen, josta voi kysyä tämän nimen osan osoitetta. Oma DNS-palvelin jatkaa kyselyä eri nimen osien DNS -palvelimilta selvittäen lopulta halutun nimen IP-osoitteen. Lopulta laite saa tältä DNS-palvelimelta halutun IP-osoitteen, johon ottaa yhteyttä. (Anttila 2000, 238.)

4 CMS MADE SIMPLE

CMS Made Simple eli Content Management System Made Simple tai lyhyemmin CMSMS on avoimen lähdekoodin sisällönhallintajärjestelmä. Ensimmäisen kerran se julkaistiin vuonna 2004. Se on PHP-pohjainen sisällönhallintajärjestelmä ja käyttää MYSQL-tietokantaa lähes kaiken sivuston sisällön tallentamiseen. CMSMS tarjoaa helpon ja yksinkertaisen tavan luoda sivustoja. CMS Made Simplen tähtäimenä onkin tarjota helppokäyttöinen mutta laajennettavissa oleva sisällönhallintajärjestelmä, jonka koko voi olla jopa sata sivua. (CMS Made Simple 2013.)

CMS Made Simplellä on selainpohjainen hallintaosio, josta voi hallita sivuston teemoja, sivujen sisältöä, sivupohjia, käyttäjiä, ryhmiä ja sivupohjien logiikkaa. Hallintasivulla pystyy myös asentamaan lisämoduulit ja hallinnoida niitä. Moduulitkin ovat avoimeen lähdekoodiin perustuvia, ja niitä tekevät CMSMS -kehittäjät ja kolmannen osapuolen vapaaehtoiset tekijät. (CMS Made Simple 2013.)

4.1 Käyttäjien roolit

CMSMS:ssä niin kuin muissakin sisällönhallintajärjestelmissä yritetään erotella sivuston kehittäjät ja muokkaajat erilleen. Sivuston kehittäjät kehittävät sivustolle sivupohjia ja logiikkaa, kun taas muokkaajat voivat näihin sivupohjiin laittaa määrättyihin paikkoihin haluttua sisältöä. (CMS Made Simple 2013.)

CMS Made Simplessä sivuston sisällön muokkaukseen ei tarvitse tietotaitoa niin paljon kuin jos se olisi luotu esimerkiksi pelkästään HTML:llä. Näin sivuston muokkaukseen ei tarvitse kouluttaa henkilöstöä niin paljon ja kehittäijien voimavaroja voidaan siirtää muihin tarkoituksiin. Tämä mahdollistaa myöskin sisällön muokkauksen asiakkaan puolelta, ja yritys voi myydä palveluna pelkästään sivuston niin sanotun alustuksen, jota asiakas pääsee sitten muokkaamaan pienellä osaamistaidolla. (CMS Made Simple 2013.)

Käyttäjien roolitus tapahtuu CMS Made Simplen hallintapaneelin kautta. Käyttäjät jaetaan ryhmiin, joilla on erilaisia oikeuksia järjestelmään. Järjestelmän

kaikille osa-alueille on mahdollisuus muokata ryhmien oikeuksia. Esimerkiksi joku käyttäjä voi luoda ja muokata sivuja mutta hänellä ei ole oikeutta koskea sivupohjiin. Käytännössä tämä tarkoittaa sitä, että tämän käyttäjän näkökulmasta koko ominaisuutta ei ole näkyvissä. Sisällönhallintajärjestelmään on siis helppo kouluttaa jokin henkilö tekstinkäsittelyosaamisella tekemään uutisia, eikä hän voi vahingossakaan koskea mihinkään, mikä rikkoisi jotain. (Hauschildt 2010, 146.)

4.2 Smarty

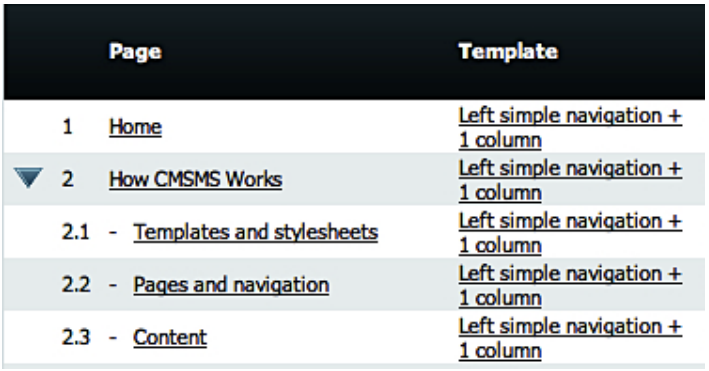
Smarty on avoimen lähdekoodin template engine PHP:lle. Sen avulla voidaan erottaa HTML:n ja CSS:n ohjelmalogiikasta, joka on tässä tapauksessa PHP. Erottamalla ohjelmalogiikka toimintalogiikasta on lähdekoodia helpompi lukea ja muokata. Samalla sivuston ulkoasua on helpompi muokata koskematta toimintalogiikkaan. (Smarty 2013.)

Smartyn yksi tärkeimmistä puolista on poistaa PHP HTML-tiedostojen sisältä ja estää kehittäjän halutessa PHP:n suorittaminen HTML-tiedostoista kokonaan. Tämä lisää turvallisuutta ja helpottaa ylläpidettävyyttä. Smartyn avulla sivupohjan tai sivun luoja ei voi siis vahingossakaan tehdä haitallista PHP-koodia ilman järjestelmänvalvojan lupaa. Tämä toisaalta tarkoittaa sitä, että sivupohjan luojan tarvitsee opetella Smartyn oma syntaksi mutta Smartyn syntaksi on myöskin paljon yksinkertaisempi opetella kuin PHP:n syntaksi. (Smarty 2013.)

4.3 Sivut ja sivupohjat

CMS Made Simplen nettiselaimella toimivassa hallintapaneelissa sivujen luontiin käytetään WYSIWYG-editointityökalua, joka tekee sivujen luomisesta yhtä yksinkertaista kuin tekstiasiakirjan tekemisen. Se poistaa sivujen luojalta tarpeen osata HTML-kieltä. CMSMS:ssä tämä editori perusasennuksessa on nimeltään TinyMCE, joka antaa laajat tekstinkäsittelytyökalut sivun luomiseen. (Hauschildt 2010, 44.)

CMSMS:ssä sivua luotaessa se sijoitetaan hierarkiaan, jonka tarkoituksena on pitää sivut halutussa järjestyksessä. Valikot CMSMS:ssä määräytyvät tämän hierarkian mukaan. Päävalikossa yleisimmässä tapauksessa näkyy vain hierarkian korkeimmat sivut kuten kuviossa 11 ”Home” ja ”How CMSMS Works” riippuen tietysti minkälaisen menun haluaa tehdä. Alisivut asettuvat pääsivujen alle hierarkiassa, ja ne yleensä laitetaan menuissa näkyviin vasta pääsivulla vieraillessa. (Hauschildt 2010, 52.)



KUVIO 11. CMS Made Simplen hallintapaneeli; sivuston sivut (Mehta 2009, 72.)

Jokaiselle sivulle asetetaan sivupohja, joka muistuttaa HTML-dokumenttia mutta siihen liimataan eri sisältömateriaaleja eri puolelta sisällönhallintajärjestelmää Smartyn komennoilla, joita kutsutaan CMSMS:ssä tageiksi. Lisäksi sivupohjiin CMSMS:ssä yhdistetään CSS-tiedostot, jotka pääsevät vaikuttamaan HTML:n ulkoasuun. (Hauschildt 2010, 292.)

Sivupohjiin voi luoda käyttäjän omatekemiä tageja, joita kutsutaan CMS Made Simplessä ”User Defined Tag” nimellä tai lyhyemmin UDT. Käyttäjien omatekemät tagit ovat kuin pieniä PHP-kielellä kirjoitettuja ohjelman osia, joita yleensä käytetään jonkin muuttujan tai attribuutin vaihtamiseen tai sitten tagin tilalle tulostamiseen. Tageille voidaan antaa parametreja, joko suoraan syöttämällä tai sitten Smartyn omia parametreja lukemalla. Käyttäjien luomat tagit ovat myöskin ainoita tageja, jotka voidaan liittää CMS Made Simplen tapahtumien hallintaan. (Hauschildt 2010, 293.)

Tapahtumien hallinnan avulla voi erilaisten tilanteiden laukaisemana ajaa tageja. Näitä tapahtumia hallitaan sivuston taustahallinnassa olevalla ”Tapahtumien

hallinta” -moduulilla. Suurin osa Eventeistä koskee hallintapaneelissa tapahtuvia asioita. Näitä ovat esimerkiksi, kun joku muokkaa, luo tai poistaa sivun sisältöä tai itse sivun. Kun jokin näistä Eventeistä laukeaa, se kutsuu siihen liitettyä käyttäjän luomaa tagia. Tämä tagi ajaa esimerkiksi CMS Mailer -moduulin, jolla se sähköpostittaa sivuston ylläpitäjälle ilmoituksen. (Hauschildt 2010, 294.)

Sivupohjien tagit korvaantuvat joko yksinkertaisella tekstillä tai jopa toisilla HTML-sivupohjilla, kun sivu ladataan selaimessa. Tageja käyttäessä useissa moduuleissa on lisäksi parametreja, joita voi syöttää kyseistä tagia käyttäessä. Esimerkiksi kuviossa 12 on tagi ”last_modified_by”, joka kertoo sivulla, kuka on viimeksi editoinut sivua. Sille määritetään ”format” parametriksi ”username”, mikä kertoo pluginille, että tieto halutaan käyttäjätunnusmuodossa. (Hauschildt 2010, 67.)

KUVIO 12. Smarty tagi CMSMS:ssä (Hauschildt 2010, 73.)

4.4 Cache

Kun sivua ladataan selaimella sivustolla vieraillessa, käyttäjälle ladataan useista eri lähteistä kasattu sivu. Koska tämä kasaus vie aikaa ja suuri osa datasta on tietokannassa, käytetään suureen osaan datasta cacheemista apuna. Cacheemisella tarkoitetaan sisällönhallintajärjestelmässä sitä, että osa tai kaikki datasta kasataan osittain tai kokonaan valmiiksi selaimelle näytettäväksi. CMS Made Simplessä Smarty kasaa tiedostot cache-kansioon, jota Smarty ylläpitää asetusten mukaan. Moduulit hallitsevat yleensä omat cachensa ja riippuu moduulin tekijästä, ovatko moduulit cachettavissa. (CMS Made Simple 2013.)

Cacheeminen CMS Made Simplessä tapahtuu monella tapaa. Sisällönhallintajärjestelmä cachee esimerkiksi sivun rakenteen tai hierarkian koostumuksen jokainen kerta kun se muuttuu. Moduulien metadata eli niiden

vaatimukset cachetaan, jotta ei tarvitse ladata kyseistä moduulia turhaan, jos sitä kutsutaan sivulla. Tyylitiedostot cachetaan selaimeen, jotta niiden lataamisessa ei tarvitse rasittaa serveriä jokaisella sivun lataamisella. Smarty tekee myöskin omia cacheja sivupohjista ja HTML-koodista nopeuttaakseen prosessointiaan. (CMS Made Simple 2013.)

5 SIVUSTON KÄYTTÖÖNOTTO

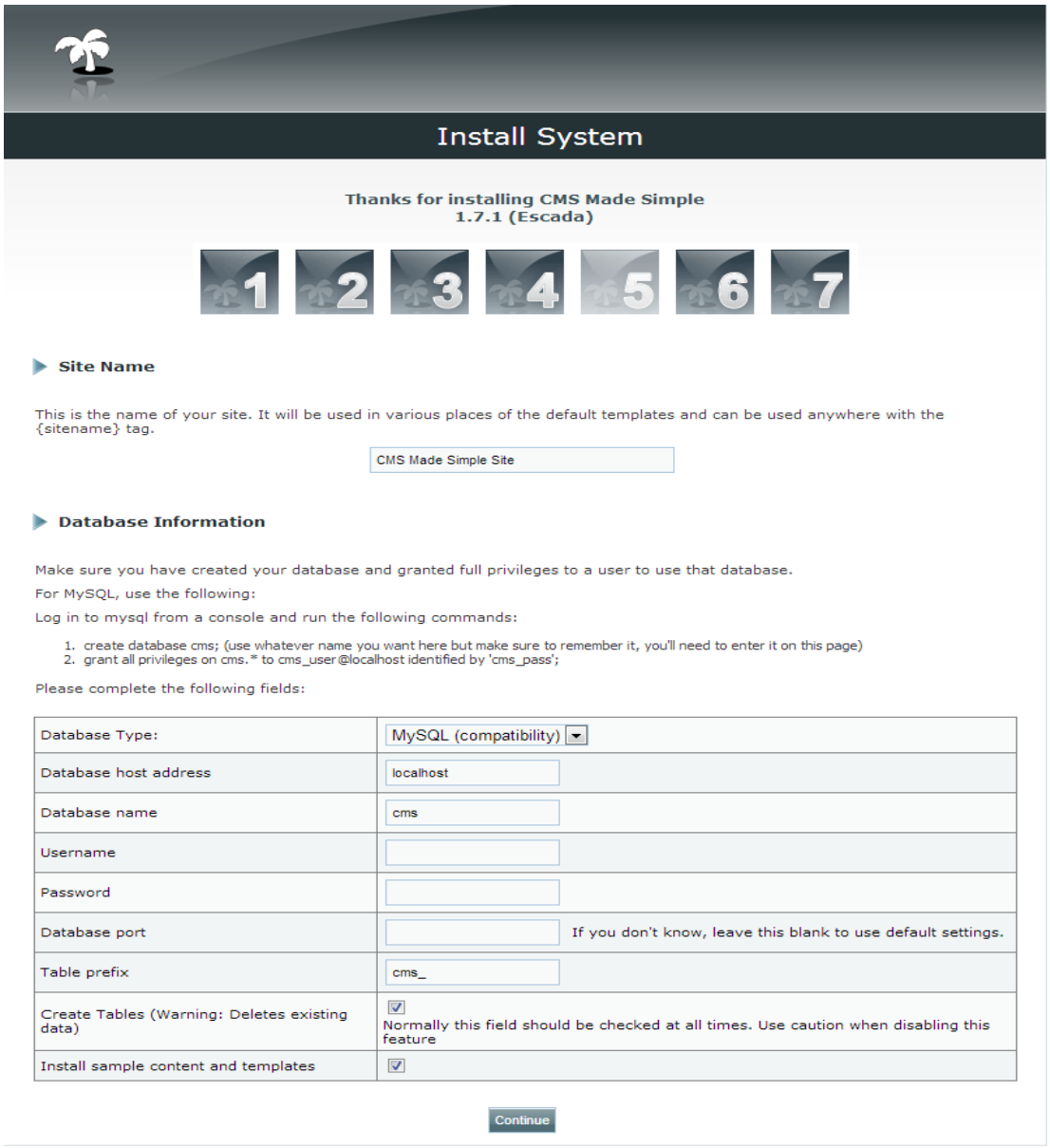
Prototyyppisivuston asennukseen hankittiin webhotellipalvelu palveluntarjoajalta. Palveluntarjoaja tarjoaa myöskin domain-palveluita, joten se sopi prototyypin tekemiseen hyvin. Palveluntarjoajalta tulee edulliseen hintaan PHP5-, SSH- ja MYSQL-palvelut nettisivujen luomiseen, ja nämä riittävät CMSMS:n tarpeisiin mainiosti. Louhi hoitaa myös lisädomainien rekisteröimiset tilauksen yhteydessä.

Aluksi asennus palvelimelle suoritetaan komentokehotetta etäkäyttämällä. Kuviossa 13 havainnollistetaan komentokehotteen komentoja. Oletetaan, että on jo navigoitu kansioon, johon CMSMS asennetaan. Tämän jälkeen ladataan asennuspaketti kyseiseen kansioon kuvion rivillä 1 olevalla komennolla. Toisen rivin komento purkaa paketin tähän samaiseen kansioon. Kolmannen rivin komento luo config.php-tiedoston ja ja antaa oikeudet lukuun ja kirjoitukseen.

```
1 wget http://s3.amazonaws.com/cmsms/downloads/5375/cmsmadesimple-1.7.1-full.tar.gz
2 tar -zxf cmsmadesimple-1.7.1-full.tar.gz
3 touch config.php; chmod 666 config.php
```

KUVIO 13. Asennuskomentoja

Kun tiedostojärjestelmä on saatu alkuasetelmiin, aletaan asentaa sisällönhallintajärjestelmää. Asennus suoritetaan käynnistämällä install.php -tiedosto selaimella install-kansiosta. Selainpohjaisesta asennuksessa on 7 vaihetta, joista erityistä huomiota vaativat admin-käyttäjän luonti vaiheessa 4 ja 5. vaiheen MYSQL-tietokannan tietojen syöttäminen, joka näkyy kuviossa 13. Tämän jälkeen vaiheessa 6 asennus kysyy vielä tietoja, joihin ei normaalisti tarvitse koskea, kuten kotihakemiston sijaintia tiedostojärjestelmässä.



KUVIO 14. CMS Made Simple -asennus selaimella

5.1 Urlille ohjaus

’.htaccess’–tiedostoilla konfiguroidaan Apache serverin kansiokohtaisia asetuksia. Tässä työssä sitä käytetään domainien ohjaamiseen alisivuille. CMS Made Simple käyttää ’.htaccess’-tiedostoja myöskin sivun osoitteen siisteyden ylläpitämiseen.

CMS Made Simplen asennuksessa tulee mukana osittain valmiiksi konfiguroitu ’.htaccess’-tiedosto. Tiedosto sijaitsee docs-kansiossa asennuksen jälkeen, ja se pitää kopioida asennuskansion juureen ja uudelleennimetä ’.htaccess’-tiedostoksi,

jotta se tulee käyttöön sivua selattaessa. Kuviossa 15 on esitetty CMS Made Simplen mukana tuleva konffi. Rivillä 4 aktivoidaan Apachen RewriteEngine, joka vaatii rivin 1 option päälle. Rivi 6 määrittelee, että kaikki tiedoston ehdot ovat relaatiossa samaiseen kansioon, jossa ’.htaccess’-tiedosto on. Rivit 8-10 tarkistavat, onko selaimen hakema polku kansio tai tiedosto, jos ei, niin se muuttaa osoitteen muotoon ”parent/child”, eli sivujen hierarkian mukaan.

```
1  Options +FollowSymLinks
2
3  <IfModule mod_rewrite.c>
4  RewriteEngine on
5
6  RewriteBase /
7
8  RewriteCond %{REQUEST_FILENAME} !-f
9  RewriteCond %{REQUEST_FILENAME} !-d
10 RewriteRule ^(.+)$ index.php?page=$1 [QSA]
11 </IfModule>
```

KUVIO 15. Asennuksen mukana tuleva .htaccess-tiedosto

Jotta sivusto toimisi usealla domainilla, muokataan jokaiselle domainille omat säännöt tähän ’.htaccess’-tiedostoon. Domainien omat ohjaukset ohjaavat sivuston juureen, joka CMS Made Simplessä on kotisivuksi asetettu sivu. Ei ole toivottavaa, että jokaisen domainin sivustolle tultaessa tullaan yhteiselle kotisivulle, koska tavoitteena on luoda yksilöllisiä kotisivuja jokaiselle loppuasiakkaalle. Nämä ’.htaccess’-ohjaukset jaetaan domainin niin sanotun pääsivun ja alisivujen osioihin. Kuviossa 16 on jokaiselle alidomainille lisättävät rivit, joista korvataan ”domain1” teksti domainien nimillä. Rivit 2-4 tarkistavat, onko pelkästään ”www.domain1.fi” kirjoitettu osoiteriville, ja ohjaa tämän kutsun ”/domain1/”-sivulle. Rivit 7-11 tarkistavat, onko jokin domainin alasivu kutsuttu, ja ohjaa sen tämän domainin alisivuille.

```
#etusivu
RewriteCond %{HTTP_HOST} ^[www.]*domain1.fi$ [NC]
RewriteCond %{REQUEST_URI} ^/$
RewriteRule ^(.*) /domain1/

#alasivut
RewriteCond %{HTTP_HOST} ^[www.]*domain1.fi$ [NC]
RewriteCond %{REQUEST_URI} !^/domain1/.*
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^([^/]+)/$ domain1/$1/ [NC]
```

KUVIO 16. .htaccess-lisäyksiä

5.2 Aliakset

Ennen kuin varsinaisia sivuja aletaan luoda, pitää ottaa huomioon, että CMS Made Simplessä jokaisella sivulla on yksilöllinen alias. Tämä alias on myöskin samalla se osoite, johon suunnistetaan selaimella sivustolla vieraillessa. Sivustoilla on kuitenkin yleensä samankaltaisia tai jopa samoja sivuja, kuten ”uutiset” tai ”yhteystiedot”. Jos käyttäjä tai jokin hakukone suunnistaa sivustolle ”domain1.fi” ja siellä valitsee menusta ”uutiset” sivulle, ei näytä hyvältä, jos osoiterivillä lukee ”domain1.fi/uutiset1” tai vastaava johtuen siitä, että joku toinen sivusto samalla asennuksella on jo vienyt ”uutiset” aliaksen.

CMS Made Simplen aliasten yksilöllisyyttä ei pysty kumminkaan kiertämään, joten sivustoja luotaessa asetetaan ”domain1”-sivuston sivuille aliakset ”domain1 -uutiset”. Näin yksilöllistetään sivustoille omat aliakset. Tämän jälkeen tehdään User Defined Tag (UDT), jota käytetään aina sivuston osoitetta hakiessa. Tämä tapahtuu linkittäessä sivuston alisivuja samalla sivustolla eli periaatteessa pelkästään menumanagerissa.

```
global $gCms;
global $smarty;

$manager =& $gCms->GetHierarchyManager();

$var = 'root_page_alias';
if( isset($params['assign']) && $params['assign'] != '' )
{
  $var = $params['assign'];
}
$result = "NO RESULT";
$thisPage = $gCms->variables['content_id'];
$currentNode = &$manager->sureGetNodeById($thisPage);
while( isset($currentNode) && $currentNode->getLevel() >= 0 )
{
    $currentContent =& $currentNode->getContent();
    $result = $currentContent->Alias();
    $currentNode =& $currentNode->getParentNode();
}
$smarty->assign($var,$result);
```

KUVIO 17. Isäntäpuun aliaksen selvittämiseen käytetty ohjelman osa

Koska osaa tähän tarkoitukseen olevasta UDT:stä käyttävät muut toiminnot CMS Made Simplessä, on tämä UDT jaettu kahteen tagiin. Näin ei tarvitse uusia koodia jälkimmäiseen tagiin vaan se käyttää ensimmäisen tagin parametriä hyväkseen. Ensimmäinen tagi on esitetty kuviossa 17, jossa smartyn funktioita käyttäen navigoidaan hierarkiapuussa kyseisen sivun juuripuuhun, jonka alias tallennetaan tagia kutsuttaessa määriteltyyn parametriin. Kuviossa 18 taas otetaan sivun parametreistä tämä kyseinen isäntäsivun alias ja sen avulla riisutaan aktiivisena olevan sivun omasta aliaksesta isäntäsivun alias ja viiva. Näin jää jäljelle esimerkiksi ”domain1-uutiset” aliaksesta pelkästään ”uutiset”-alias.

```
global $gCms;
$root_page_alias = $gCms->smarty->get_template_vars('root_page_alias');
$var = str_replace($root_page_alias.'-','',$params[alias]);
echo $var;
```

KUVIO 18. Isäntäpuun aliaksen riisuminen

5.3 Menu Manager

Menut CMS Made Simplessä käsitellään yleensä Menu Managerissa, joka tulee asennuksen mukana coremoduulina. Menu Manager ottaa sivuhierarkian käsittelyyn ja navigoi sitä silmukassa oikeassa järjestyksessä riippuen siitä, miten Managerin templatet on luotu. Ilman muokkauksia usealla domainilla Menu Manager ei osaa pitää kiinni domain-nimestä osoitekentässä vaan linkittäisi CMS Made Simplen asetusten mukaiseen domainiin.

Edellä mainituista syistä lisätään aiemmin luodut tagit linkkien luontivaiheeseen lisäten vielä CMS Made Simplen oma tagi. Tämä tagi kertoo urlin, joka on tällä hetkellä voimassa. Kuviossa 19 on karkeasti esitetty Menu Manageriin lisätyt rivit. Kuviossa havainnollistetaan vain CMS Made Simplen multidomainiksi muokkaamista varten tarvittavat rivit. Kuviossa rivillä 1 haetaan domainin nimi CMS Made Simplen omalla tagilla ja tallennetaan se muuttujaan. Rivillä kaksi kutsutaan sivulla 22 kuviossa 17 esitetyn kaltaista tagia isäntäpuun aliaksen hankkimiseksi. Rivillä kolme käytetään näitä tietoja luoden linkitys sen hetkiselle menun elementille. Tähän tarkoitukseen käytetään tagia, joka riisuu domainin nimen kyseisen sivun aliaksesta, jos kyseessä ei ole isäntäpuu. Tämä tehdään siksi, että vaikka periaatteessa käyttäjältä on piilossa osoitteet, johon linkit ohjaavat näkyy se osoiterivillä. Muuten yksinkertaisissa sivuston osoitteissa kuten ”www.domain1.fi/uutiset” olisi ylimäärästä esimerkkinä ”www.domain1.fi/domain1-uutiset” .

```
1 {capture assign='root_url'}{root_url}{/capture}
2 {get_root_page_alias assign ="root_page_alias"}
3 <a href="{root_url}/{strip_sub_page_alias alias=$node->alias}"><span>{$node->menutext}</span></a>
```

KUVIO 19. Menu Manageriin lisätyt rivit

5.4 Sivupohja ja sivujen yleismuotoilu

Tässä työssä tehdyillä muokkauksilla CMS Made Simpleen on mahdollisimman pienellä sivupohjien määrällä mahdollista luoda toimiva sisällönhallintajärjestelmä. Jotta sivustojen asennus ja käyttöönotto olisi mahdollisimman yksinkertaista, CSS-asetuksissa värien ja optioiden hallintaan

tarvitaan logiikkaa. Tähän on kaksi ratkaisua CMS Made Simplen 1.7 versiossa. Sivuston luoja voi joko tehdä jokaiselle sivustolle oman CSS-tiedoston sisällönhallintajärjestelmässä tai käyttää jotain tiedostoa tai tietokantaa osalle ulkonäköasetuksista. Uuden CSS-tiedoston ja sivupohjan luomisessa on kuitenkin suurempi työ, ja koska sivustojen erot ovat kuitenkin vain taustavärien luokkaa, päädytään käyttämään tietokantaa.

Tähän tarkoitukseen CMS Made Simplessä ei ole suoraa työkalua, ja oman moduulin luominen olisi tuskin kannattavaa. Tämän takia luodaan tietokantaan taulu, jossa määritellään CSS:ään soveltuvalla tavalla väriarvoja. Tämä lisää serverivaatimuksia niin, että serverille tarvitsee asentaa phpMyAdmin ohjelma. Tämä helpottaa sitä, että sivuston ylläpitäjältä vaadita MYSQL-osaamista vaan hän pystyy toimimaan selaimella myöskin siinä ympäristössä.

TAULUKKO 1. Sivustojen väriarvoja MYSQL-tietokannassa

←T→	alias	palautesposti	bodytausta	headertausta	contenttausta	footertausta	menutausta	menuactfontvari	menuinactfontvari	menuhoverfontvari
		NULL	red		green	blue	brown	NULL	NULL	NULL
	oy-yritys-ab	NULL	NULL	NULL	NULL	NULL	brown	green	grey	black

Taulukossa 1 on esitetty phpMyAdminin näkymästä MYSQL-taulukkoa. Tässä taulukossa on parille sivustolle määritetty arvot. Näitä arvoja luetaan tagilla, joka on esitetty kuviossa 20. Tässä tagissa otetaan yhteys tietokantaan smartyn funktioilla. Sitten siinä määritellään isäntäpuun alias silmukassa kuin sivulla 22 kuviossa 17 luodussa tagissa. Tätä aliasta käytetään taulusta tiedon hakemiseen. Tagia kutsuttaessa on otettava huomioon, että sille pitää antaa parametrina haettavan tiedon nimi-, assign- ja echo-arvo. Jos echo- tai assign-arvoa ei ole annettu, tagi ei tee mitään. Echo-arvon ollessa asetettuna tagi palauttaa kyseisen arvon sen sivupohjassa kutsuttuun paikkaan. Tämä mahdollistaa sen, että tagin voi laittaa haluttuun väliin, jossa se joko korvaantuu haetulla arvolla tai sitten sitä voidaan käyttää jonkin parametrin luomiseen.

```
global $gCms;
global $smarty;
$db = &$gCms->db;
if (! $db )
    die ("Ei voitu yhdistää MySQL");
$manager =& $gCms->GetHierarchyManager();
$var = $params['nimi'];
$var2 = $params['assign'];
$echotus = 0;
if(isset($params['echo']))
    $echotus = $params['echo'];
$result = "NO RESULT";
$thisPage = $gCms->variables['content_id'];
$currentNode = &$manager->sureGetNodeById($thisPage);
while( isset($currentNode) && $currentNode->getLevel() >= 0 )
{
    $currentContent =& $currentNode->getContent();
    $result = $currentContent->Alias();
    $currentNode =& $currentNode->getParentNode();
}
$q = "SELECT ".$var." FROM yritysrekisteri WHERE alias = '".$result."'";
$dbresult = $db->Execute($q);
if( !$dbresult )
    echo 'DB error: '. $db->ErrorMsg()."<br/>";
else{
    $arr = $dbresult->FetchRow() ;
if(isset($var) && $arr[$var] != NULL && $echotus != 0)
        echo $arr[$var];
if(isset($var2) && $arr[$var] != NULL)
  $smarty->assign($var2,$arr[$var]);
}
```

KUVIO 20. HaeYritysRekisteristä tagi

5.5 Tyylitykset ja kuvat

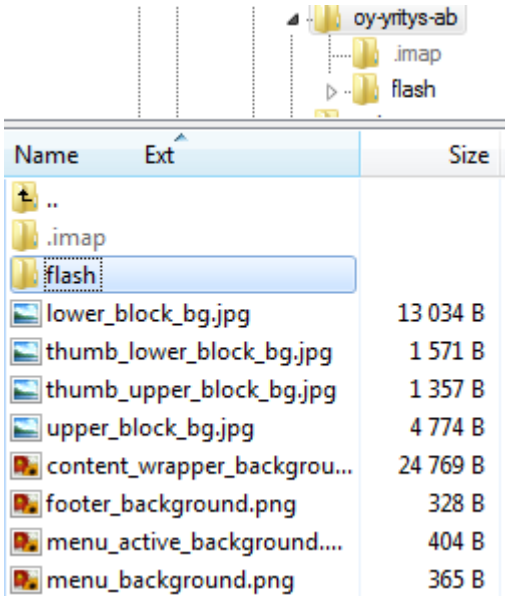
Normaalisti CSS-tyylitiedostot liitettäisiin CMSMS:ssä määrittelemällä ne sivupohjaan hallintapaneelin kautta ja sivupohjassa niitä kutsuttaisiin Smarty-tagin avulla. Tähän CMSMS tarjoaa oman tagin ”{stylesheets}”. Tämä tagi prosessoi tyylitiedostot ja tekee niistä muun muuassa cachet prosessoinnin yhteydessä. Koska tyylitiedostoihin ei ole CMS Made Simplen 1.7.1 versiossa mahdollista laittaa minkäänlaista logiikkaa, on useiden sivujen jakamassa sivupohjassa pakko irrottaa osa tyylitiedoston määrittelyistä itse sivupohjaan.

Koska Smarty käyttää aaltosulkeita omien tagien ja komentojen tunnistamiseen, Smarty tarjoaa ’{literal}’ tagin, joka lopettaa Smartyn prosessoinnin ’{/literal}’ tagiin asti. Tämä auttaa, kun sivupohjassa on CSS-tyylityksiä tai javascriptiä, koska nekin käyttävät aaltosulkeita omiin tarkoituksiinsa. Kuviossa 21 on esitetty osa sivupohjan tyylittelyistä, joihin on lisätty logiikkaa sivun 25 kuvion 20 tagilla. Tagia käytetään tässä osassa lähinnä tekstien värityksen muokkaamiseen sivukohtaisesti mutta sitä voi käyttää missä tahansa sivupohjan kohdassa tyylittelemään HTML-elementtejä tietokannasta määritellyillä arvoilla.

```
<style type="text/css">
    ul#primary-nav li a.menuactive {
        color: {/literal}{HaeYritysasetuksista echo='1' nimi='menuactfontvari'}{literal};
        font-weight: bold;
    }
    ul#primary-nav li a.menuactive:hover {
        color: {/literal}{HaeYritysasetuksista echo='1' nimi='menuhoverfontvari'}{literal};
        font-weight: bold;
    }
    ul#primary-nav li a {
        font-size: 1em;
        font-weight: normal;
        color: {/literal}{HaeYritysasetuksista echo='1' nimi='menuinactfontvari'}{literal};
        padding: 11px 15px 11px;
        display: block;
        text-decoration: none;
    }
    ul#primary-nav li a:hover {
        color: {/literal}{HaeYritysasetuksista echo='1' nimi='menuhoverfontvari'}{literal};
        font-weight: normal;
        font-size: 1em;
        text-decoration: none;
        display: block;
```

KUVIO 21. Tyylit sivupohjassa logiikalla varustettuna

Monessa tilanteessa tyylityksissä on myös kuvia käytössä. Näihin tapauksiin on sovittava nimeämiskäytäntö, koska aliakset ovat ainoa tapa löytää nämä kuvat muiden joukoista luomatta jokaiselle sivustolle omaa sivupohjaa. Jokaiselle sivustolle luodaan oma kansio yhteiseen images-kansioon. Tämä kansio nimetään sivuston aliaksen mukaan, jotta kuvia osataan etsiä oikeasta kansiosta. Kuvien nimet ovat ennaltamääriteltyjä riippuen siitä, mihin niitä tullaan käyttämään. Esimerkiksi ”banner.jpg” tulee banneriin jne. Kuviossa 22 on esitetty oy-yritys-ab -sivuston images-kansiota. Kuvat on nimetty niiden sijoitusten mukaan. Esimerkiksi ”footer_background.png” tulee automaattisesti sivuston footerin taustakuvaksi.



KUVIO 22. Esimerkkikuvia sivuston omassa kansiossa

Kuviossa 23 on luotu tagi, joka etsii näistä kansioista halutut tiedostot. Kuviosta on poistettu alusta osa, joka löytyy sivulta 25 kuvion 20 alusta ja joka koskee aliaksen hakemista hierarkiasta. Tagilla kuviossa 23 etsitään sivuston aliaksen kansiosta kuva ja asetetaan se parametriin. Jos kuvaa ei löydy kansiosta tai kansiota ei ole käytetty, käytetään default-kuvaa, koska sivupohjat on rakennettu niin, että jokaiselle sivulle on pakko tehdä näihin paikkoihin jonkinlaiset kuvat.

```
1  $var = '';
2  if( isset($params['assign']) && $params['assign'] != '' )
3  {
4    $var = $params['assign'];
5  }
6  $kuva = '';
7  if( isset($params['kuva']) && $params['kuva'] != '' )
8  {
9   $kuva = $params['kuva'];
10 }
11
12 if(isset($kuva) && $kuva != ''){
13 $result2 = "uploads/images/" .$result ."/" .$kuva;
14 $result3 = "uploads/images/" .$kuva;
15 $onkoJokin = false;
16 $paatteet = array('.jpg','.JPG','.png','.PNG','.gif','.GIF');
17 foreach($paatteet as $paate){
18     if(file_exists($result2 .$paate)){
19         $result2 .= $paate;
20         $smarty->assign($var,$result2);
21         $onkoJokin = true;
22     }
23 }
24 if(!$onkoJokin){
25     foreach($paatteet as $paate){
26         if(file_exists($result3 .$paate)){
27             $result3 .= $paate;
28             $smarty->assign($var,$result3);
29         }
30     }
31 }
32 }
```

KUVIO 23. Kuvien hakeminen images-kansiosta

Monessa tilanteessa kuvien koko kuitenkin vaihtelee, joten tarvitaan jonkinlainen tapa saada sekin yksilöityä ilman, että sivupohjien määrä kasvaisi liikaa. Tähän tagit antavat keinon tehdä PHP-tarkistuksia kuvan koosta ja määritellä sivun elementtejä dynaamisesti sivuston tarpeiden mukaan. Kuviossa 23 esitetään kuvan koon selvittämistä PHP-koodilla CMS Made Simplen tagissa. Tagille annetaan parametrina kuvan nimi, jonka olemassaolo tarkistetaan rivillä 5. Kuvatiedostoa tutkitaan sen jälkeen PHP-funktiolla ’getimagesize’ rivillä 8. Kuvan korkeus tulostetaan tagin tilalle. Tätä tagia käytetään lähinnä HTML div -elementtien korkeuksien määrittelemiseksi sivupohjassa, jotta kuvat asettuvat hyvin kuvan kokoon.

```
global $gCms;
global $smarty;

$kuvaurl = '';
if(isset($params['kuva'])&& $params['kuva'] != '')
{
    $kuvaurl = $params['kuva'];
    if($kuva = getimagesize($kuvaurl)){
    //list($width, $height, $type, $attr) = getimagesize($kuvaurl);
    list($width,$height,$type,$attr) = $kuva;
    echo 'height:'.$height.'px;';
    }
```

KUVIO 24. Kuvan korkeuden määrittely tagissa

5.6 Sivut

▼	3	domain1	Yritys_etusivu	Content
	3.1	- Etusivu		Internal Page Link
	3.2	- Tuotteet	Yritys_alisivu	Content
	3.3	- Palaute	Yritys_alisivu	Content
▼	4	Domain2	Yritys_etusivu	Content
	4.1	- Etusivu		External Link
	4.2	- Tuotteet	Yritys_etusivu	Content
	4.3	- Liiketilat	Yritys_etusivu	Content
▼	5	domain3	Yritys_etusivu	Content
	5.1	- Etusivu		External Link
	5.2	- Ruokalistat	Yritys_alisivu	Content
	5.3	- Näin löydät perille	Yritys_alisivu	Content
	5.4	- Kuvia ravintoloista	Yritys_alisivu	Content

KUVIO 25. Sivustoja sivunäkymässä

Kun kaikki esivalmistelut on tehty, voidaan rakentaa sivustoja sivunäkymässä sisällönhallintajärjestelmän hallintasivulla. Kuviossa 21 on demonstroitu haluttua hierarkiarakennetta. Hierarkiapuun korkeimmaksi elementiksi asetetaan sivuston pääsivu, jolle on luotu Yritys_etusivu-sivupohja. Menujen takia luodaan sisällönhallintajärjestelmän sisäinen linkki heti sen alle. MenuManageri käsketään

sivupohjissa näyttämään vain toisen asteen sivut, kun se luo menut sivuille. Tällöin se näyttää vain sivuston omat linkit.

Sivupohjissa pitää ottaa huomioon niiden standardisoiminen. Yhden sivupohjan pitää soveltua useaan sivustoon käytettäväksi. Sivupohjan HTML head tagien osiossa määritellään yleensä sivuston otsikkoarvo, joka kerrotaan kävijälle yleensä selaimen ikkunan otsikkona. CMS Made Simplessä on sen mukana tulevissa ja yleisesti käytetyimmissä sivupohjamalleissa sivun otsikon arvo määritelty Smartyn tageilla ja sen muoto on kuin kuviossa 26. Tagein kerrotaan sivun oma otsikko, joka muokataan sivun omissa asetuksissa, ja sen jälkeen kerrotaan sivuston nimi. ’{sitename}’ tagin tilalle voidaan tehdä käyttäjän luoma tagi, joka kertoo sivuston aliaksen alkuosan muokattuna kuten edellisissä esimerkeissä on useiden tagien sisällä tehty.

```
1  <title>{title} | {sitename}</title>
```

KUVIO 26. Sivuston otsikkokenttä

6 YHTEENVETO

CMS Made Simplen muokkaaminen yhden sivuston hallintajärjestelmästä monen sivuston hallintajärjestelmäksi saatiin toteutettua pH Kolmen vaatimuksiin vastaavaksi. Sivustojen luominen oli muokkausten jälkeen helppoa ja vaivatonta palautteen mukaan. Sisällönhallintajärjestelmään on tullut myöhemmin monta uutta päivitystä, jotka helpottavat tämän opinnäytetyön sisällön toteuttamista. Ajanpuutteen vuoksi tätä ei kuitenkaan ole toteutettu uusimmilla versioilla.

CMS Made Simplen multidomain-muokkausta käytti pitkään yksi pH Kolmen asiakas, ja he saivat monta hienoa sivustoa aikaiseksi. Monia asioita, jotka päätyivät tähän opinnäytetyöhön, muokattiin heidän palautteensa mukaan.

Jälkikäteen ajateltuna olisi kannattanut vaihtaa sisällönhallintajärjestelmää, kun halusi usealla domainilla toimivan järjestelmän. Olisi kuitenkin ollut yritykselle kalliimpaa opetella ja opettaa asiakkaille uusia järjestelmiä, jos he olisivat halunneet kyseisen kaltaisia järjestelmiä. Aikataulullisesti tämän työn tekemiseen meni pari kuukautta. Aikaa on kuitenkin vaikea arvioida muiden projektien yhtäaikaisuuden takia. Henkilökohtainen PHP-osaaminen vahvistui ja sisällönhallintajärjestelmien parissa työskentely tehostui huomattavasti tämän projektin parissa.

Usealla domainilla toimiva ja tällä tavoin muokattu sisällönhallintajärjestelmä ei ole tällä hetkellä toiminnassa missään. Tästä projektista tuli kuitenkin monta hyvää palautetta. Toivottavasti tulevaisuudessa joku ottaa käyttöön usean domainin sisällönhallintaratkaisuja ja käyttää tämän projektin pohjaa apunaan.

LÄHTEET

Anttila, A. 2001. TCP/IP Tekniikka. Helsinki Media.

CMS Made Simple 2013. CMS Made Simple Official Documentation. CMS Made Simple [viitattu 21.3.2013]. Saatavissa: http://docs.cmsmadesimple.org

CMS Matrix. 2013. The Content Management Comparison Tool. Plain Black Corporation [viitattu 6.3.2012]. Saatavissa: http://www.cmsmatrix.org/matrix/cms-matrix

Drupal Installation. 2013. Drupal Community Documentation. Drupal [viitattu 18.4.2013]. Saatavissa: http://drupal.org/documentation/install

Mehta, N. 2009. Choosing an Open Source CMS: Beginners Guide. Packt Publishing.

Smarty. 2013. About Smarty. New Digital Group [viitattu 21.3.2013]. Saatavissa: http://www.smarty.net/about_smarty

Hauschildt, S. 2010. CMS Made Simple 1.6: Beginner's Guide. Packt Publishing.

Wikipedia, 2012a. Sisällönhallintajärjestelmä. Wikimedia-säätiö [viitattu 9.10.2012]. Saatavissa: http://fi.wikipedia.org/wiki/Sis%C3%A4ll%C3%B6nhallintaj%C3%A4rjestelm%C3%A

Wikipedia, 2012b. Web content management system. Wikimedia-säätiö [viitattu 9.10.2012]. Saatavissa: http://en.wikipedia.org/wiki/Web_content_management_system

Wikipedia, 2012c. WWW-sisällönhallinta. Wikimedia-säätiö [viitattu 9.10.2012]. Saatavissa: http://fi.wikipedia.org/wiki/Www-sis%C3%A4ll%C3%B6nhallinta